(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 May 2006 (26.05.2006)

PCT

(10) International Publication Number
WO 2006/055445 A2

(51) International Patent Classification:
G06F 12/14 (2006.01)

(21) International Application Number:
PCT/US2005/041024

(22) International Filing Date:
14 November 2005 (14.11.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/627,502	13 November 2004 (13.11.2004)	US
60/628,517	15 November 2004 (15.11.2004)	US

(71) Applicant *(for all designated States except US)*: **STREAM THEORY, INC.** [US/US]; Suite 350, 19600 Fairchild Road, Irvine, CA 92612 (US).

(72) Inventors: **DE VRIES, Jeffrey. ZIVERTNIK, Greg. HUBBLE, Ann.**

(74) Agent: **AHMANN, William**; Perkins Coie LLP, 101 Jefferson Drive, Menlo Park, CA 94025 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2006/055445 A2

(54) Title: HYBRID LOCAL/REMOTE STREAMING

(57) Abstract: A technique for streaming from a local device and a remote device involves providing a subset of data associated with a streaming application to a local device. An example of a method according to the technique includes running a streaming software player, accessing data from a local storage device necessary to stream a software application, and accessing data from a remote location necessary to stream the software application using the streaming software player. An example f a system according to the technique may include, by way of example but not limitation, a means for streaming a first subset of content associated with a streaming application from a streaming server, a means for streaming a second subset of content associated with the streaming application from a local source, and a means for periodically querying the streaming server, wherein the periodic querying informs the streaming server that the streaming application is running.

HYBRID LOCAL/REMOTE STREAMING

BACKGROUND

Software streaming involves downloading small pieces of files as the pieces are needed by the program being streamed. These small pieces may be referred to as blocks. A streaming client sends requests for blocks as they are needed up to a streaming server, which sends back streaming data that is associated with the requested block. Sending a request and receiving the streaming data may cause delays that can slow down the streamed program.

On advantage of streamed applications is that users only download what is needed to execute a program. The provider of a streamed application has more control over the streamed application than, for example, a provider of a program that is installed to hard disk. For example, the provider of a streamed application may charge a per hour rate for the use of the application. Alternatively, the provider of a streamed application may prevent a local disk from being filled up unnecessarily with code that will not be accessed.

While streaming has certain advantages, there are many problems associated with streaming software that it would be advantageous to negate, work around, or reduce. For example, speeding up execution of streaming programs is an on-going issue. As another example, it may be desirable to provide a user with an experience similar to that associated with an installed, as opposed to streamed, program. As another example, it may be desirable to provide streamed programs on consoles.

In addition, a provider of a streamed application may desire to attempt copyright protection. This can be difficult because if a streamed application is downloaded entirely to a local machine, the user of the local machine may gain access to all portions of the streamed application, eliminating the need to stream and increasing the risks of copyright violations. Thus, an additional concern for some may be to prevent a user from gaining access to all of a given program locally.

SUMMARY

A technique for streaming from a local device and a remote device involves providing a subset of data associated with a streaming application to a local device. By way of example but not limitation, a first subset of the data may be provided to a consumer on a removable storage device. An example of a method according to the technique includes running a streaming software player, accessing data from a local storage device necessary to stream a software application, accessing data from a local cache necessary to stream the software application, and accessing data from a remote location necessary to stream the software application using the streaming software player. In a non-limiting embodiment, the method may include sending a request to stream the software application. The software application may then be streamed from both the local storage device and the remote, for example, streaming server.

Another example of a method according to the technique includes maintaining, at a streaming server, first content associated with a streaming application, receiving a request to stream the streaming application, and providing a token file in response to the request. The token file may include, by way of example but not limitation, an address associated with the streaming server, instructions to receive the first content from the streaming server, and instructions to receive second content from a local streaming source, wherein said first content and said second content are necessary for properly streaming the streaming application. In a non-limiting embodiment, the method may include dividing content associated with a streaming application into the first content and the second content.

An example of a system according to the technique may include, by way of example but not limitation, a means for streaming a first subset of content associated with a streaming application from a streaming server, a means for streaming a second subset of content associated with the streaming application from a local source, and a means for periodically querying the streaming server, wherein the periodic querying informs the streaming server that the streaming application is running.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention are illustrated in the figures. However, the embodiments and figures are illustrative rather than limiting; they provide examples of the invention.

FIG. 1 depicts a flowchart of a process for streaming a software title and other activities associated with the software title.

FIGS. 2A to 2N are intended to illustrate an example of a streaming software experience.

FIGS. 3A and 3B depict systems for hybrid local/remote streaming.

FIGS. 4A and 4B depict examples of a system for hybrid local/networked streaming.

FIG. 5 depicts a flowchart of a method for hybrid local/remote software streaming.

FIG. 6 depicts a flowchart of a method for hybrid local/remote software streaming.

FIG. 7 depicts a networked system for use in an embodiment.

FIG. 8 depicts a computer system for use in the system of FIG. 7.

DETAILED DESCRIPTION

A technique for a "streaming install" of a software title involves reducing many of the steps typically required to run the software, such as installation of the software and configuration of the environment. A goal when streaming a title is providing a user experience after startup that closely resembles that of an installed software title.

An advantage of an aspect of the technique is that users can experience faster and easier startup than is typical for installation of non-streaming software titles. Another advantage is that PC games, for example, can be made as quick and easy to play as console games. A measure of performance in a streaming application is whether the streamed application performs virtually the same as the same title installed traditionally (i.e., not streamed). In some cases, streamed titles have even performed better than traditionally installed titles.

Some users believe that installation on a hard disk can improve streaming software title performance. This may be true since a CD or DVD may spin down at certain points and incur the delay of spinning up again. Accordingly, in an embodiment, streaming applications are installed on a hard disk. It should be noted that installation of a streaming application on a hard disk is not the same as installation of a non-streaming application; the streaming application is still streamed after installation, it is just streamed from the hard disk instead of, for example, a CD or DVD.

Since it may be desirable to support streaming from multiple disks, a full installation can be used to reduce the problem of having to swap disks. In some cases, it may be desirable to require that a single CD or DVD be available as a key, such as a Macrovision SafeDisc key. This may be valuable as a piracy-prevention measure.

Similarly, streamed applications may support applications utilizing Macrovision SafeDisc for security. For example, a CD key may be verified at startup an periodically through the use of the software title using Macrovision SDK features. The Macrovision SafeDisc key may be on each CD.

Some users may wish to modify ("mod") a software title, which requires access to assets and files associated with the software title. Accordingly, users may be provided with access to source assets provided with the software title, such as art assets (models, textures, sounds, level

files, etc.) These can be opened modified, copied, or inserted locally. Similarly, users may be able to download a mod from a network and integrate the mod into the software title.

Users and software title providers typically also find software patches desirable. A streamed application may provide a method for users to download the latest patches.

As is common for most software titles in the industry, product registration may be desired when providing CD or DVD streaming titles.

Compression techniques may be applied to fully utilize media on which streamed data files are provided. For example, packaging of data into streaming buffers may be as compact as Zip-style compression.

FIG. 1 depicts a flowchart of a typical process for streaming a software title, and other activities associated with the software title. In a non-limiting embodiment, at module 102 a user puts a CD into a computer, as illustrated in FIG. 2A. In a non-limiting embodiment, upon putting the CD into the computer, an "Autorun.inf" file is processed by the operating system (e.g., Windows 2000) of the computer. The Autorun.inf file references and starts a "Stub" program. At module 104, the Stub program provides initial options to the user.

The Stub program may provide features, such as product registration, streaming player (e.g., Stream Weaver) detection, streaming player installation, providing the user with various menu options, etc. Product registration may involve determining whether a CD registration key has been entered by checking persistent memory on the user's computer. In an embodiment, if the product has not been registered (typically, this is true the first time a title is streamed), then a registration program prompts the user for the relevant registration information, as illustrated in FIG. 2B.

In an embodiment, the Stub program may provide a screen for entering the CD registration key and product registration. The Stub program may save information about the CD registration key and product registration in persistent memory. Registration information may be sent to a vendor or producer of the software title over a network (e.g., the Internet). If no network connection is available, registration may be handled in some other manner, as is well-understood in the art of software registration.

If the user skips the registration process, this fact may be saved in persistent memory, and used when deciding whether to ask the user to register again later. If the registration has been

previously declined, then this fact may be saved in persistent memory and a check for whether a predetermined time period has passed is made. If the time period has passed, then the user is given the chance to register again.

In another embodiment, if a CD has been previously registered, the Stub program skips registration and goes directly to a splash screen, as described later with reference to FIG. 2D.

As previously indicated, the Stub program may provide features such as streaming player detection and streaming player installation, as illustrated in FIG. 2C. These features may involve checking for the installation and, if installed, the version of a streaming player by checking the registry for a relevant key value. For example, the registry may include the key value "HKEY_LOCAL_MACHINE\SOFTWARE\Stream Theory\Stream Theory Player\InstallPath". If the key is present, that should mean the Stream Theory Player is installed and identify the location of the installation. The version of the installed player may be determined by checking the version of the file specified in the key, typically being "C:\Program Files\Stream Theory\stfsd.sys" with a key value of 4.0.0.20. If the player is found, then the version of the player should be compared against the minimum player required. If no player is installed, or if the version is insufficient to support a desired software title, the player, which may be available for install from the CD on with the title is provided, or from a remote location, such as a web site. Installation of the new player may require a reboot.

In a non-limiting embodiment, at module 106 (FIG. 1), the Stub program may also provide the user with various options on a "splash screen," such as is illustrated in FIG. 2D. The options may include an autoplay option 108 (FIG. 1), an install-to-hard-drive option 110 (FIG. 1), a file editor option 112 (FIG. 1), and an upgrade option 114 (FIG. 1). Additional options (not shown) may include an option to visit a web site, such as the web site of the vendor or producer of the software title, or to quit.

If the user clicks on the autoplay option, as illustrated in FIG. 2E, at module 116 (FIG. 1) the user is given the opportunity to begin receiving the streaming application right away. For example, the game shell may launch, as illustrated in FIG. 2F, presenting user with ability to load game, play new game, set options, etc. Behavior is similar to that of a traditionally-installed software application, and is not necessarily limited to games.

At module 118 (FIG. 1) the software title streams data from the CD (or hard drive, if fully installed previously), as illustrated in FIG. 2G. In addition, Macrovision Safedisc may be

used to check keys or other data, as necessary. At optional module 120 (FIG. 1) the executable checks Macrovision Safedisc. The file system must be intelligent enough to identify on which disc desired files are located, prompting the user to swap discs if necessary at module 122 (FIG. 1). If installed to hard drive, when the file system requests another CD, the user may, for example, enter a key sequence to go on. In this example, the key sequence may be intercepted by a driver that lets the game proceed as if another disc was inserted. Other techniques may also be used to provide multi-disc functionality.

At module 124 (FIG. 1), if the user clicks on the install-to-hard-drive option, as illustrated in FIG. 2H, the user if given the opportunity to install the streaming application onto a (usually) local hard drive. The techniques involved in handling an installed streaming application and a streaming application running from multiple CDs are similar. Each CD is equivalent to a stream file. When the application prompts for a particular CD, this is detected by requests for blocks outside the range of the currently active stream file. The streaming player would have to intercept and detect requests to change CDs.

When installing a streaming title on the hard drive, the system typically checks for adequate disk space and copies all of the streaming files to the hard drive, if space is sufficient, as illustrated in FIG. 2I. In addition, shortcuts may be created at optional module 126 (FIG. 1), window registry settings may be adjusted, and removal code for the short cuts may be added to the "Add and Remove" programs. Prompting of the user to load the CDs in a certain order may be required. Once a streaming title has been installed onto the hard drive, the user may select, for example, a shortcut to launch the streaming player and run the streaming application, as illustrated in FIG. 2J. The next time "Play Now" is selected, the cache on the hard drive can be used, as indicated at module 128 (FIG. 1).

At module 112 (FIG. 1), if the user clicks on the file-editor option, as illustrated in FIG. 2K, a file editor is launched at module 130 (FIG. 1), as illustrated in FIG. 2L. The user may gain access to specific files that can be modified or replaced. Normally, the file structure of a streamed application is hidden from the user. If an application modifies a file, this file is copied out a subdirectory, such as the directory "C:\Stream Theory\local\app_name\...\file_to_be_edited".

As with application patching, the editor should be processed with the streamed application and included in the same stream file as the streamed application so that the files are

visible from within the editor. When the modified file is saved, it will not go to the stream file but will be saved under the directory specified above. The editor may modify and write out files that are to be made visible to external applications. Data files may not be "unpackable" so the streaming data files may not be extracted from them in an offline manner.

At module 114 (FIG. 1), if the user clicks on the upgrade option, as illustrated in FIG. 2M, an upgrade program launches to check for and apply updates at module 132 (FIG. 1), as illustrated in FIG. 2N. Updates may include upgrades, patches, or other changes. With a streaming application patch, the upgrade Stub-program may invoke a token file that starts the patch program. The patch program may then go out onto the Web to find and retrieve any new patches. If there is a new patch, this file may be downloaded and saved to a local directory, for example, "C:\Program Files\Streamtheory\local\app_name\...". The streaming player's file system driver may check the local application directory for files that may have been updated or for files that are to be replaced with newer updates. As the application executes, the patched file will be accessed and visible to the user.

The patch program should be streamed and processed with the streaming application. This may be required if, for example, the application file structure is hidden from outside applications. File hiding may be done for security reasons, to prevent unauthorized copying of an application.

When a streaming application expects content on a CD or DVD in, for example, a CD or DVD drive, the user may be required to swap in the appropriate disc. The streaming application itself may include a token file that is encoded with information related to data that resides on a particular CD or DVD. In order to reduce the problems with making sure the proper disc is available, the CD or DVD drive can be virtualized.

FIG. 3A depicts a system 300 for hybrid local/remote streaming. The system 300 includes a first streaming content database 302, a second streaming content database 304, a local cache 306, and a streaming player 308. The first streaming content database 302 is located at a streaming server 310. The second streaming content database 304 is located at a streaming client 312.

In an embodiment, the second streaming content database 304 is included on a removable media, such as a CD or DVD, that is distributed to a consumer. The second streaming content may be streamed directly from the removable media or installed in local memory or non-volatile

storage, such as a hard disk drive. The removable media may include copy protection safeguards of types that are well-known in the art and are continuously being developed to protect against privacy and other forms of intellectual property theft.

In an embodiment, the second streaming content database 304 does not include all of the content associated with a streaming application. The first streaming content database 302, however, includes the rest of the streaming content for the streaming application. As is known in the art of content streaming, the local cache 306 may be utilized while the streaming player 308 runs the streaming application. When content, such as a block, is requested from the first streaming content, the streaming player requests the block from the streaming server 310. On the other hand, when content, such as another block, is requested from the second streaming content, the streaming player requests the block from a local source, such as the removable media on which the second streaming content database 304 was provided, or a local source, such as a hard drive onto which the second streaming content database 304 was installed. A request may be honored by the local cache 306 if the requested content, such as a block, is cached.

Advantageously, the system 300 can be used to stream software at a high rate of speed, even if broadband access is unavailable or limited, because some of the requested blocks are available locally. For large software titles, this can be of great value. Moreover, it is sometimes valuable to keep certain data relatively secure. The data that should be kept secure can be encoded into the blocks that are provided locally.

FIG. 3B depicts an example of the system 300 that includes a heartbeat monitor 330. The heartbeat monitor 330 is occasionally pinged, queried, or otherwise notified of the ongoing streaming of a streaming application. It may be desirable to know how much a software title is used, and for how long, for the purposes of, for example, charging for the time spent using the software title. The heartbeat monitor 330 may receive additional information, as well, such as which blocks are being requested and in what order, that may help to predictively provide replies to block requests or to configure content databases in future versions of the software title.

FIG. 4A depicts an example of a system 400 for hybrid local/networked streaming. The system 400 includes a streaming player 402, a first content database 404, a second content database 406, and a network interface 408. The streaming player 402 and the first content database 404 are on a streaming client 412, which is coupled to a network 420, which may

include the Internet. The second content database 406 and the network interface 408 are located on a streaming server 410, which is also coupled to the network 420.

In an embodiment, the first content database 404 is missing every 20th block and the second content database 406 includes every 20th block. When the streaming player 402 runs a streaming application associated with the blocks in the first and second content databases 404, 406, the streaming player 402 makes block requests of the databases. If a block request is for one of the blocks in the first content database 404, the block request is honored locally. If, on the other hand, the block request is for one of the blocks in the second content database 406, then the block request is made via the network 420. The request is honored by the streaming server 410, which sends a reply to the block request through the network interface 408 via the network 420 to the streaming player. It should be noted that once a request has been honored by the streaming server 410, the requested block may be cached locally.

The streaming application may be configured to hold any of a variety of different sets of blocks (e.g., every 5th block, every block that is smaller than a given size, etc.). FIG. 4B depicts an alternative configuration of the content databases for the system 400. In the example of FIG. 4B, blocks associated with a streaming application are organized into libraries. The content database located on the streaming server 410 includes the first 5 blocks of each library, depicted in FIG. 4B as library block subset 416-1 to 416-N for libraries 1 to N, respectively. The content database located on the streaming client 412 includes the 6th and later blocks of each library, depicted in FIG. 4B as library block subset 414-1 to 416-N for the libraries 1 to N, respectively.

Advantageously, arranging the content databases as depicted in FIG. 4B allows for a download period during which blocks are requested from the streaming server 410, followed by streamed content from a local (probably faster) source. In an embodiment, the blocks located in the streaming server 410 include header information that occurs during natural breaks in a software title. For example, a game that has multiple levels may be configured such that header information is requested from the streaming server 410 prior to beginning each level, then, during the level, content is requested locally. This may entail a pause between levels, depending upon the predictive request capabilities, bandwidth, and other factors.

FIG. 5 depicts a flowchart 500 of a method for hybrid local/remote software streaming. In an embodiment, the flowchart 500 starts with streaming 502 a first subset of content associated with a streaming application from a streaming server. In an embodiment, the

flowchart 500 continues with streaming 504 a second subset of content associated with the streaming application from a local source. In an embodiment, the flowchart 500 continues with periodically querying 506 the streaming server, wherein the periodic querying informs the streaming server that the streaming application is running.

FIG. 6 depicts a flowchart 600 of a method for hybrid local/remote software streaming. In an embodiment, the flowchart 600 starts with dividing 602 content associated with a streaming application into first content and second content. This may be conducted in a streamification stage of streaming software development. Alternatively, a previously streamified application may be divided according to an algorithm (e.g., removing every 20th block from a first streaming content database and placing every 20th block in a second streaming content database). In an embodiment, the flowchart 600 continues with maintaining 604 the first content on a streaming server. This may entail providing a web site to which users may visit to begin streaming an application. The web site may be at the same physical location of the streaming content, or a remote location. In an embodiment, the flowchart 600 continues with receiving 606 a request to stream the streaming application. In an embodiment, the flowchart 600 continues with providing 608 a token file in response to the request, the token file including:

an address associated with the streaming server;
instructions to receive the first content from the streaming server; and
instructions to receive the second content from a local streaming source.

The streaming software player used to stream applications, in an embodiment, may make use of a DLL that is used to determine if an FSD has been loaded or if a user has been loaded. The user should be able to stream from multiple stream files. Common files of an application must fit on one CD. The user should be able to intercept requests for new discs or the virtual equivalent for an installed to hard drive software title switching stream files.

For files installed on the hard drive, the registry should be checked for the streaming software player to determine if streaming is to be done from a CD, from the stream files stored on the hard disk, or from somewhere else. If it is desired for a streaming application to run like software that has been fully installed on disk, then that streaming application should know when it is to switch to a different stream file. The user file system driver may support a system level call to switch between stream files for this purpose. Fully installed applications, may make an

application call to the FSD to change. For patch support, a user may be required to support a command line switch to invoke different executables. For software title editor support, the logic for when files are written out to a directory associated with the streaming software player, as distinguished from the software title, should be handled carefully.

A streaming application may be configured to enable reading SafeDisc data. The streamed application may be configured so that it appears as if the application is entirely installed on the C: drive whether it is running off CD or the hard drive. The problem occurs if part of the application appears to be installed on the hard drive and part on the CD. If this is the case, this fact may be captured during application processing and stored in a token file. When the application makes a request from the CD, the file system driver may intercept the request and determine if the request is valid for the application. If the request is for SafeDisc information, then the interception of the CD request by the ST FSD may cause this request to fail.

However, if the application is organized (everything appears on the C: drive) so that no data remains on the CD in the installed state, during the processing of the application this fact may be indicated in the token file. In this case, the file system driver will not intercept the calls to the CD and requests for the SafeDisc information may succeed.

This might seem confusing because where the application thinks the files are located and where the actual data is streamed from are two different things. The actual stream data could be coming from CD, the network, or the hard disc where in all these cases the files and executables all appear to be on the C: drive. Nevertheless, when the installed version of the application is transformed into a streaming format, the application does not assume that some of the data is on the CD (excluding SafeDisc data).

For a multi-CD streamed application, there are specific requirements imposed on how the code is developed and structured. In a non-streamed multi-CD game, as new files are needed that are located on different CDs, the application requests that a particular CD be inserted. The old CD is unmounted and the new CD is mounted upon CD insertion. For a streamed multi-CD game, the behavior may be somewhat different. To a streamed application, the entire directory structure for all the files appear to be present even though the actual files are not on a currently mounted CD. This is because the entire application file and directory structure has to be known at the time the application is streamified.

This causes a problem if the application tries to reference a part of the application that is not present because the CD with the stream data blocks is not present. Even though the read file appears to be available, the read will fail. Because of this, the application has this structural dichotomy that it appears as if all the files are present but the application has to be aware of when it needs to request that a different CD be inserted. Like in a non-streamed multi-CD game, the game puts up a request for the new CD to be inserted. This also implies that each CD can't have an overlapping directory structure.

Streamed applications may be structured so that the parts of the application that are common between CDs are known. The streamed application can lock these common parts into the cache or copy them to the hard drive. These files may be identified at the time the application is turned into a streamed application.

Online game play may proceed similarly to a game accessed through a CD or through a CD with formatted stream data. The access time to retrieve executables or data for a game that was traditionally executed from a CD is virtually identical to the time required to access data from a CD with stream data. This is also true for applications saved to disk. Additionally, given in this situation that none of the executables or data are accessed over the network, there is no or minimal competition for bandwidth between on-line playing data exchanges and the accesses to the hard disk or CD.

The game file system may be modified to search the hierarchy of local, hard drive, and streamed CD caches. Files may have a disc (cache) identifier. The file system may prompt the user to insert the correct disc, if necessary. The game should limit disc swapping points for simplicity; this is likely on level boundaries (at least), so the swapping happens in the user interface shell, not the running gameplay. This directory could be pre-computed before hand and be an additional data file on the CD. Handling multiple discs will require duplicated data on each disc; this could impact overall media space requirements negatively. Possible alternatives to address this are to always do a partial install to the hard disc, installing shared components rather than duplicating them in the stream theory buffers on separate discs.

The following description of FIGS. 7 and 8 is intended to provide an overview of computer hardware and other operating components suitable for performing the methods of the invention described herein, but is not intended to limit the applicable environments. Similarly, the computer hardware and other operating components may be suitable as part of the

apparatuses of the invention described herein. The invention can be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention can also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network.

FIG. 7 depicts a networked system 700 that includes several computer systems coupled together through a network 702, such as the Internet. The term "Internet" as used herein refers to a network of networks which uses certain protocols, such as the TCP/IP protocol, and possibly other protocols such as the hypertext transfer protocol (HTTP) for hypertext markup language (HTML) documents that make up the World Wide Web (the web). The physical connections of the Internet and the protocols and communication procedures of the Internet are well known to those of skill in the art.

The web server 704 is typically at least one computer system which operates as a server computer system and is configured to operate with the protocols of the world wide web and is coupled to the Internet. The web server system 704 can be a conventional server computer system. Optionally, the web server 704 can be part of an ISP which provides access to the Internet for client systems. The web server 704 is shown coupled to the server computer system 706 which itself is coupled to web content 708, which can be considered a form of a media database. While two computer systems 704 and 706 are shown in FIG. 7, the web server system 704 and the server computer system 706 can be one computer system having different software components providing the web server functionality and the server functionality provided by the server computer system 706, which will be described further below.

Access to the network 702 is typically provided by Internet service providers (ISPs), such as the ISPs 710 and 716. Users on client systems, such as client computer systems 712, 718, 722, and 726 obtain access to the Internet through the ISPs 710 and 716. Access to the Internet allows users of the client computer systems to exchange information, receive and send e-mails, and view documents, such as documents which have been prepared in the HTML format. These documents are often provided by web servers, such as web server 704, which are referred to as being "on" the Internet. Often these web servers are provided by the ISPs, such as ISP 710, although a computer system can be set up and connected to the Internet without that system also being an ISP

Client computer systems 712, 718, 722, and 726 can each, with the appropriate web browsing software, view HTML pages provided by the web server 704. The ISP 710 provides Internet connectivity to the client computer system 712 through the modem interface 714, which can be considered part of the client computer system 712. The client computer system can be a personal computer system, a network computer, a web TV system, or other computer system. While FIG. 7 shows the modem interface 714 generically as a "modem," the interface can be an analog modem, isdn modem, cable modem, satellite transmission interface (e.g. "direct PC"), or other interface for coupling a computer system to other computer systems.

Similar to the ISP 714, the ISP 716 provides Internet connectivity for client systems 718, 722, and 726, although as shown in FIG. 7, the connections are not the same for these three computer systems. Client computer system 718 is coupled through a modem interface 720 while client computer systems 722 and 726 are part of a LAN 730.

Client computer systems 722 and 726 are coupled to the LAN 730 through network interfaces 724 and 728, which can be Ethernet network or other network interfaces. The LAN 730 is also coupled to a gateway computer system 732 which can provide firewall and other Internet-related services for the local area network. This gateway computer system 732 is coupled to the ISP 716 to provide Internet connectivity to the client computer systems 722 and 726. The gateway computer system 732 can be a conventional server computer system.

Alternatively, a server computer system 734 can be directly coupled to the LAN 730 through a network interface 736 to provide files 738 and other services to the clients 722 and 726, without the need to connect to the Internet through the gateway system 732.

FIG. 8 depicts a computer system 740 for use in the system 700 (FIG. 7). The computer system 740 may be a conventional computer system that can be used as a client computer system or a server computer system or as a web server system. Such a computer system can be used to perform many of the functions of an Internet service provider, such as ISP 710 (FIG. 7).

In the example of FIG. 8, the computer system 740 includes a computer 742, I/O devices 744, and a display device 746. The computer 742 includes a processor 748, a communications interface 750, memory 752, display controller 754, non-volatile storage 756, and I/O controller 758. The computer system 740 may be couple to or include the I/O devices 744 and display device 746.

The computer 742 interfaces to external systems through the communications interface 750, which may include a modem or network interface. It will be appreciated that the communications interface 750 can be considered to be part of the computer system 740 or a part of the computer 742. The communications interface can be an analog modem, isdn modem, cable modem, token ring interface, satellite transmission interface (e.g. "direct PC"), or other interfaces for coupling a computer system to other computer systems.

The processor 748 may be, for example, a conventional microprocessor such as an Intel Pentium microprocessor or Motorola power PC microprocessor. The memory 752 is coupled to the processor 748 by a bus 760. The memory 752 can be dynamic random access memory (DRAM) and can also include static ram (SRAM). The bus 760 couples the processor 748 to the memory 752, also to the non-volatile storage 756, to the display controller 754, and to the I/O controller 758.

The I/O devices 744 can include a keyboard, disk drives, printers, a scanner, and other input and output devices, including a mouse or other pointing device. The display controller 754 may control in the conventional manner a display on the display device 746, which can be, for example, a cathode ray tube (CRT) or liquid crystal display (LCD). The display controller 754 and the I/O controller 758 can be implemented with conventional well known technology.

The non-volatile storage 756 is often a magnetic hard disk, an optical disk, or another form of storage for large amounts of data. Some of this data is often written, by a direct memory access process, into memory 752 during execution of software in the computer 742. One of skill in the art will immediately recognize that the terms "machine-readable medium" or "computer-readable medium" includes any type of storage device that is accessible by the processor 748 and also encompasses a carrier wave that encodes a data signal.

Objects, methods, inline caches, cache states and other object-oriented components may be stored in the non-volatile storage 756, or written into memory 752 during execution of, for example, an object-oriented software program. In this way, the components illustrated in, for example, FIGS. 1-3 and 6 can be instantiated on the computer system 740.

The computer system 740 is one example of many possible computer systems which have different architectures. For example, personal computers based on an Intel microprocessor often have multiple buses, one of which can be an I/O bus for the peripherals and one that directly connects the processor 748 and the memory 752 (often referred to as a memory bus). The buses

are connected together through bridge components that perform any necessary translation due to differing bus protocols.

Network computers are another type of computer system that can be used with the present invention. Network computers do not usually include a hard disk or other mass storage, and the executable programs are loaded from a network connection into the memory 752 for execution by the processor 748. A Web TV system, which is known in the art, is also considered to be a computer system according to the present invention, but it may lack some of the features shown in FIG. 8, such as certain input or output devices. A typical computer system will usually include at least a processor, memory, and a bus coupling the memory to the processor.

In addition, the computer system 740 is controlled by operating system software which includes a file management system, such as a disk operating system, which is part of the operating system software. One example of an operating system software with its associated file management system software is the family of operating systems known as Windows® from Microsoft Corporation of Redmond, Washington, and their associated file management systems. Another example of operating system software with its associated file management system software is the Linux operating system and its associated file management system. The file management system is typically stored in the non-volatile storage 756 and causes the processor 748 to execute the various acts required by the operating system to input and output data and to store data in memory, including storing files on the non-volatile storage 756.

Some portions of the detailed description are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to

these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention, in some embodiments, also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the methods of some embodiments. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language, and various embodiments may thus be implemented using a variety of programming languages.

While this invention has been described in terms of certain embodiments, it will be appreciated by those skilled in the art that certain modifications, permutations and equivalents thereof are within the inventive scope of the present invention. It is therefore intended that the following appended claims include all such modifications, permutations and equivalents as fall within the true spirit and scope of the present invention; the invention is limited only by the claims.

CLAIMS

What is claimed is:

1. A method, comprising:

running a streaming software player;

accessing data from a local storage device necessary to stream a software application using the streaming software player;

accessing data from a local cache necessary to stream the software application using the streaming software player; and

accessing data from a remote location necessary to stream the software application using the streaming software player.

2. The method of claim 1, further comprising sending a request to stream the software application.

3. The method of claim 1, further comprising receiving in response to a request to stream the software application a token file.

4. The method of claim 1, further comprising receiving in response to a request to stream the software application an address of a streaming server at the remote location.

5. The method of claim 1, further comprising receiving in response to a request to stream the software application instructions to access first content from the remote location and to access second content from the local storage device.

6. The method of claim 1, further comprising periodically querying a streaming server at the remote location, wherein the periodic querying informs the streaming server that the streaming application is running.

7. The method of claim 1, wherein the local storage device is a removable storage device.

8. A method, comprising:

maintaining, at a streaming server, first content associated with a streaming application;

receiving a request to stream the streaming application; and

providing a token file in response to the request, said token file including:

an address associated with the streaming server;

instructions to receive the first content from the streaming server; and

instructions to receive second content from a local streaming source, wherein said first content and said second content are necessary for properly streaming the streaming application.

9. The method of claim 8, further comprising dividing content associated with a streaming application into the first content and the second content.

10. The method of claim 8, further comprising maintaining the first content on a networked server.

11. The method of claim 8, further comprising providing a file necessary to stream the streaming application to a consumer, wherein the file includes a local address associated with the second content and a remote address associated with the second content.

12. The method of claim 8, further comprising placing the second content on a removable media.

13. The method of claim 8, further comprising making available to a consumer removable media having the second content stored thereon.

14. A system, comprising:

a means for streaming a first subset of content associated with a streaming application from a streaming server;

a means for streaming a second subset of content associated with the streaming application from a local source; and

a means for periodically querying the streaming server, wherein the periodic querying informs the streaming server that the streaming application is running.

15. The system of claim 14, further comprising a means for running a streaming software player.

16. The system of claim 14, further comprising a means for sending a request to stream the software application to the streaming server.

17. The system of claim 14, further comprising a means for accessing data from a local storage device necessary to stream a software application.

18. The system of claim 14, further comprising a means for accessing data from a local cache necessary to stream the software application.

19. The system of claim 14, further comprising a means for accessing data from a remote location necessary to stream the software application using the streaming software player.

20. The system of claim 14, wherein the local source is a removable storage device.

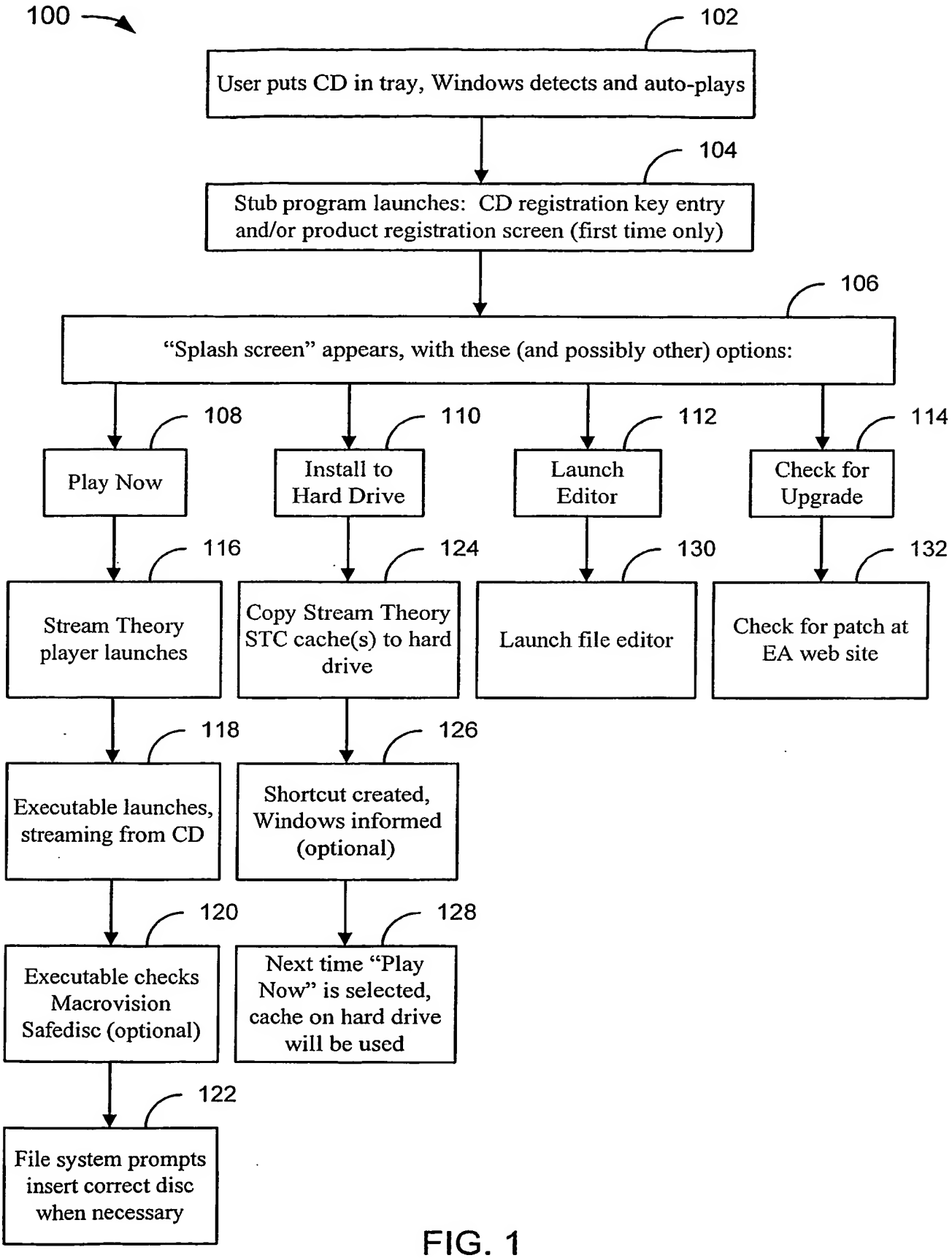

100
102
User puts CD in tray, Windows detects and auto-plays
104
Stub program launches: CD registration key entry and/or product registration screen (first time only)
106
"Splash screen" appears, with these (and possibly other) options:
108
Play Now
110
Install to Hard Drive
112
Launch Editor
114
Check for Upgrade
116
Stream Theory player launches
124
Copy Stream Theory STC cache(s) to hard drive
130
Launch file editor
132
Check for patch at EA web site
118
Executable launches, streaming from CD
126
Shortcut created, Windows informed (optional)
120
Executable checks Macrovision Safedisc (optional)
128
Next time "Play Now" is selected, cache on hard drive will be used
122
File system prompts insert correct disc when necessary

FIG. 1

FIG. 2A

The *CD Streaming* Experience

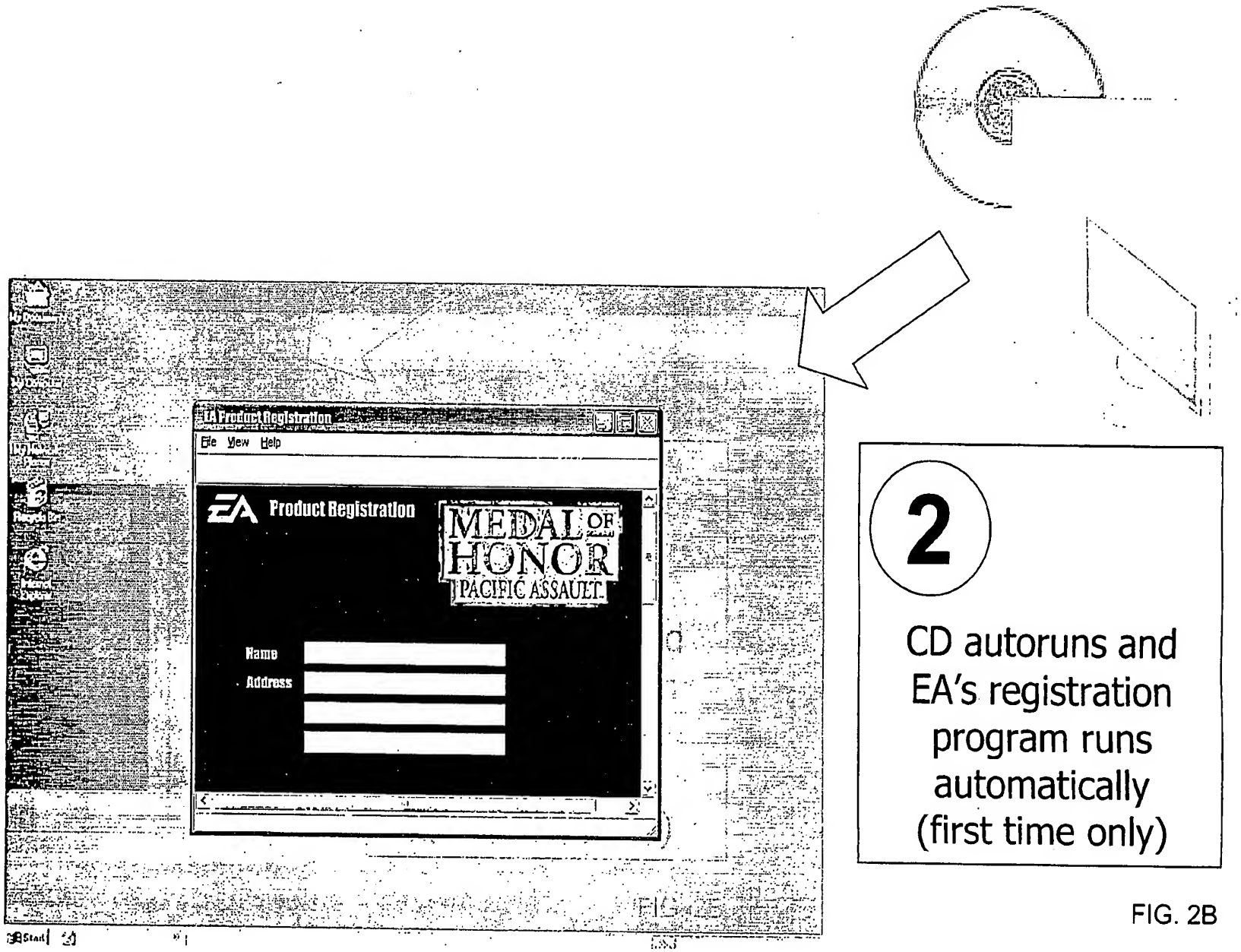


FIG. 2B

The CD Streaming Experience

3 Stream Theory Player gets automatically installed (if not already installed)

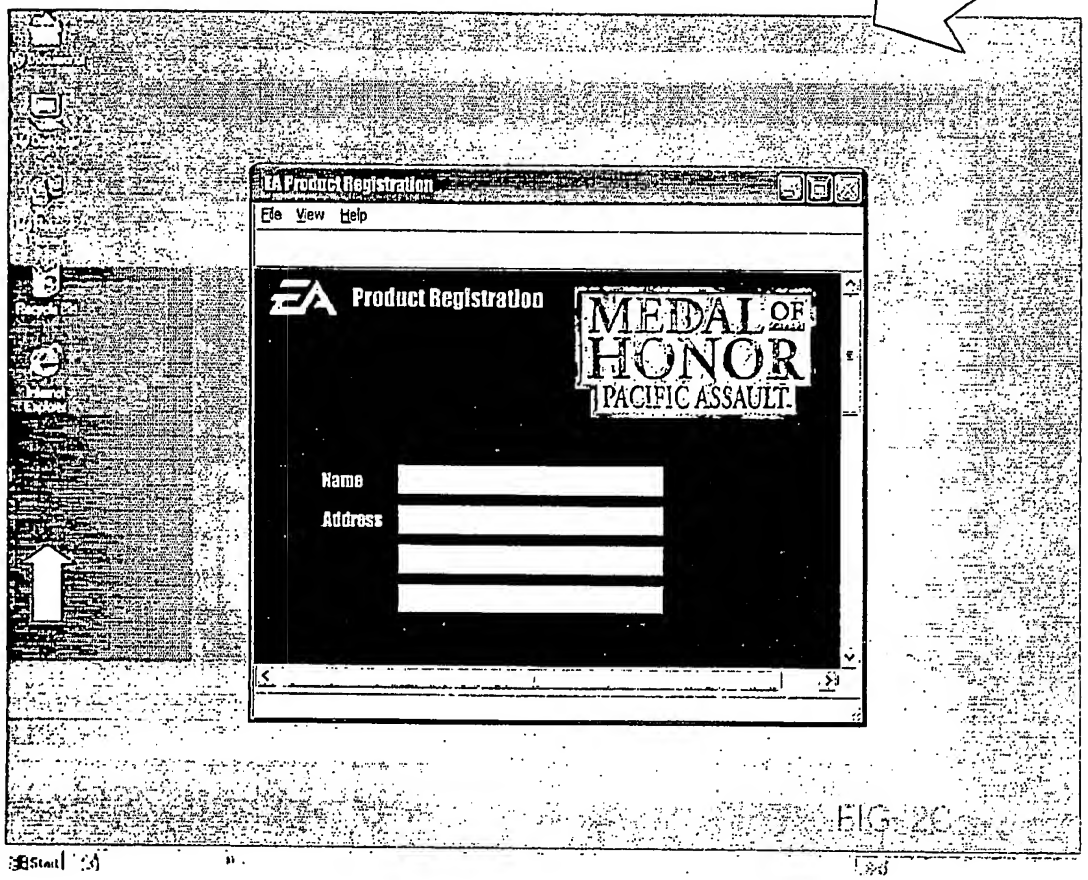
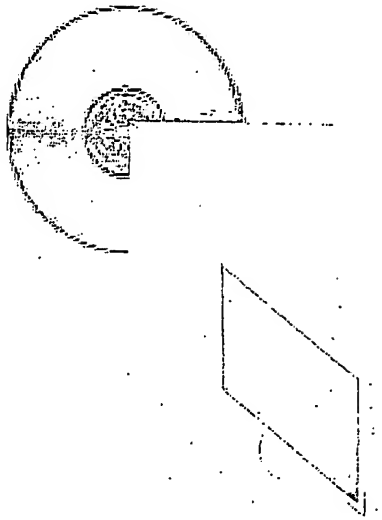


FIG. 2C

The *CD Streaming* Experience

4 **Splash Screen is Shown**



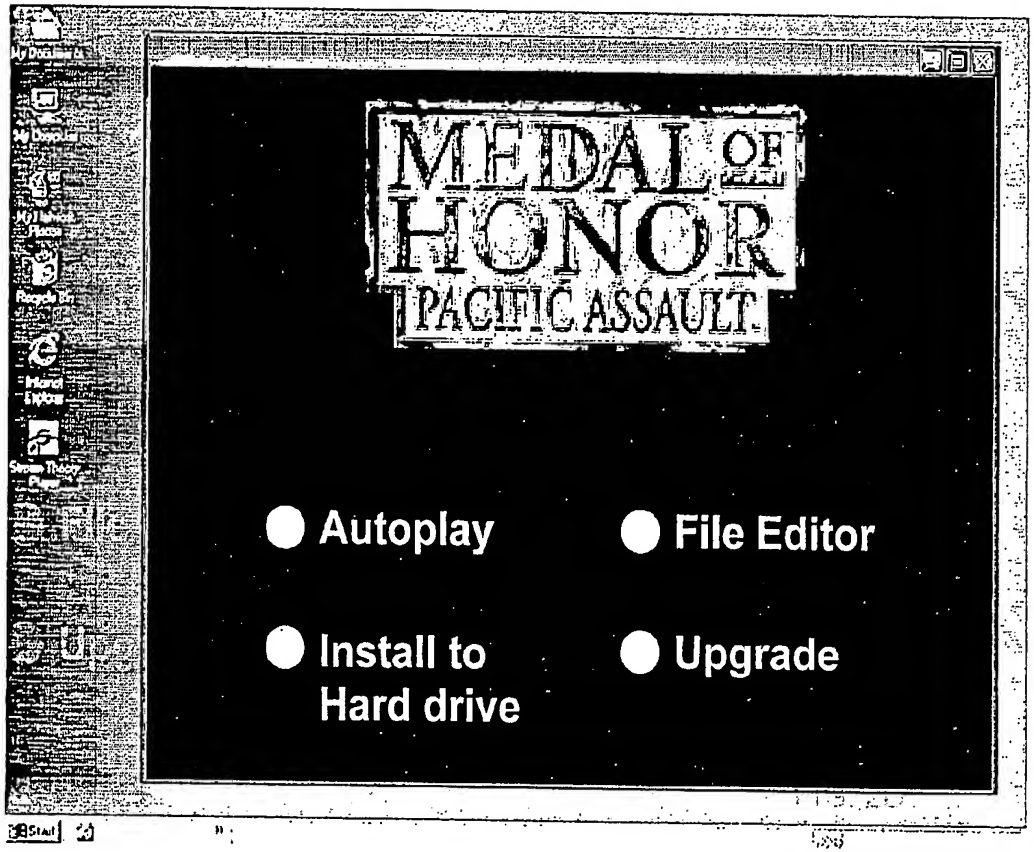
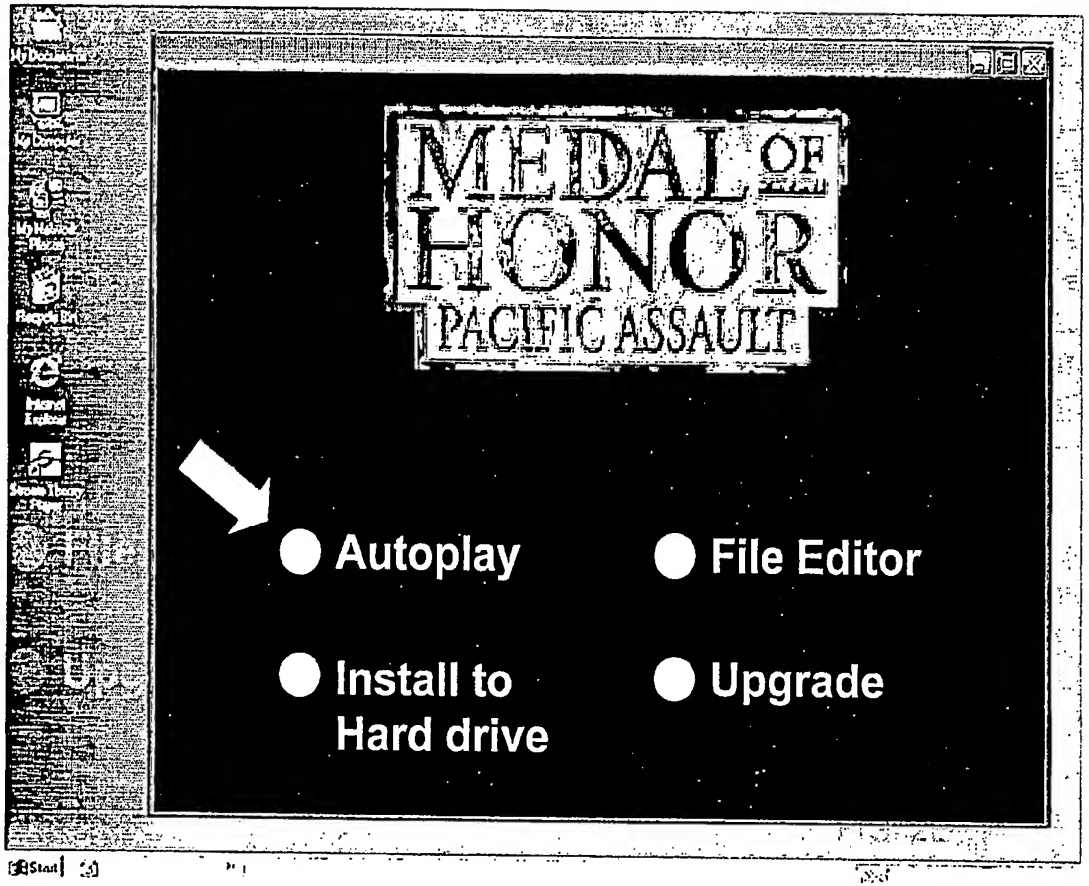


FIG. 2D

The *CD Streaming* Experience

Click on "Autoplay"



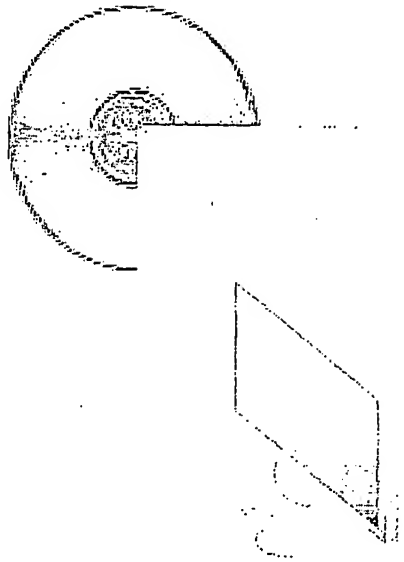
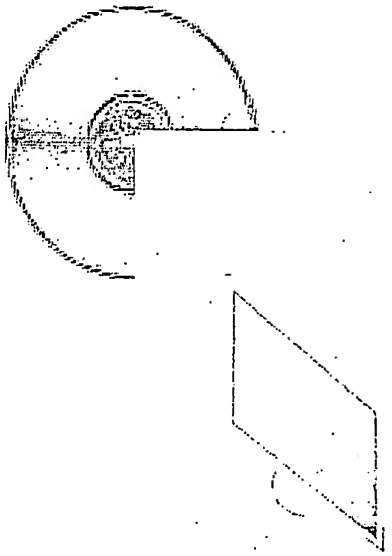
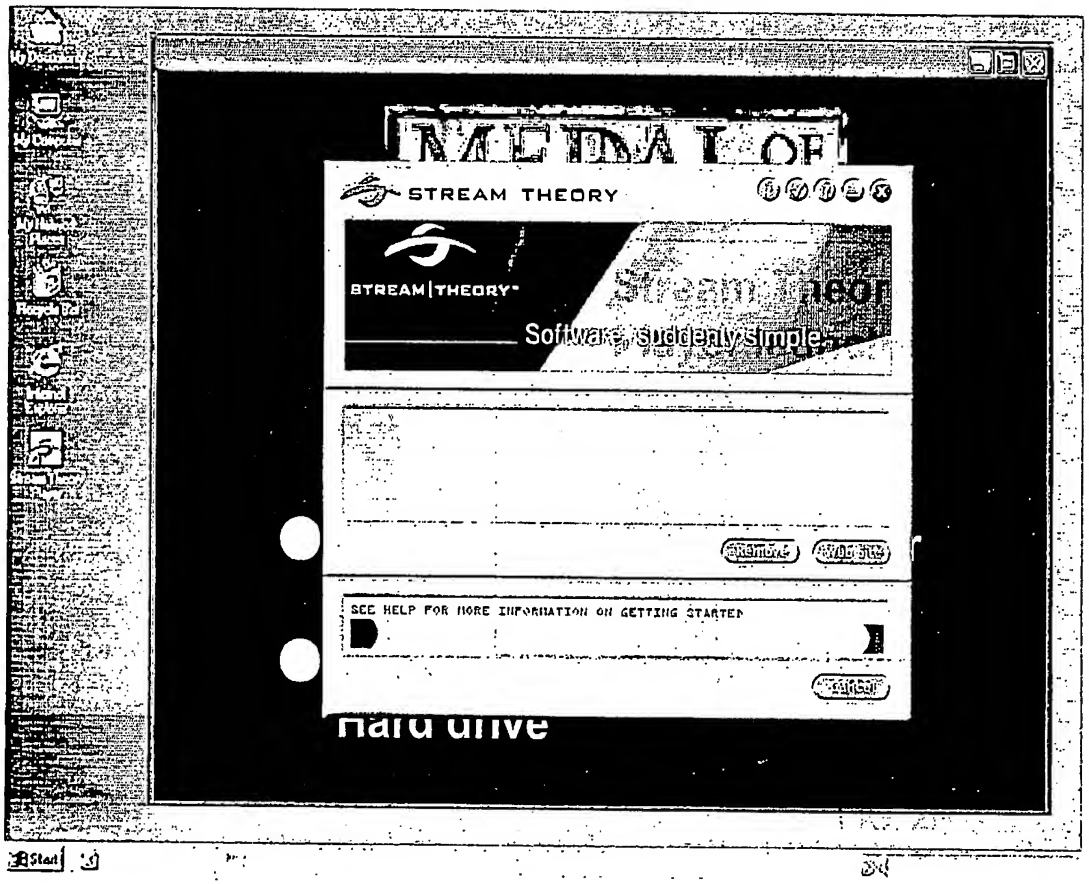


FIG. 2E

The CD Streaming Experience

Click on "Autoplay"



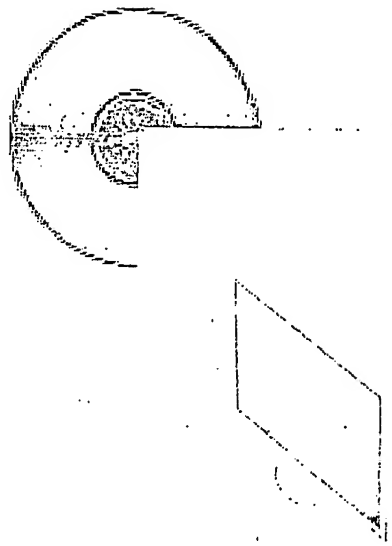


Stream Theory Player launches

FIG. 2F

The *CD Streaming* Experience

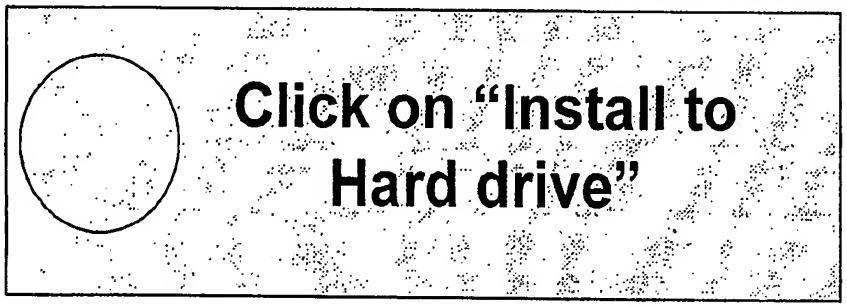
Click on "Autoplay"

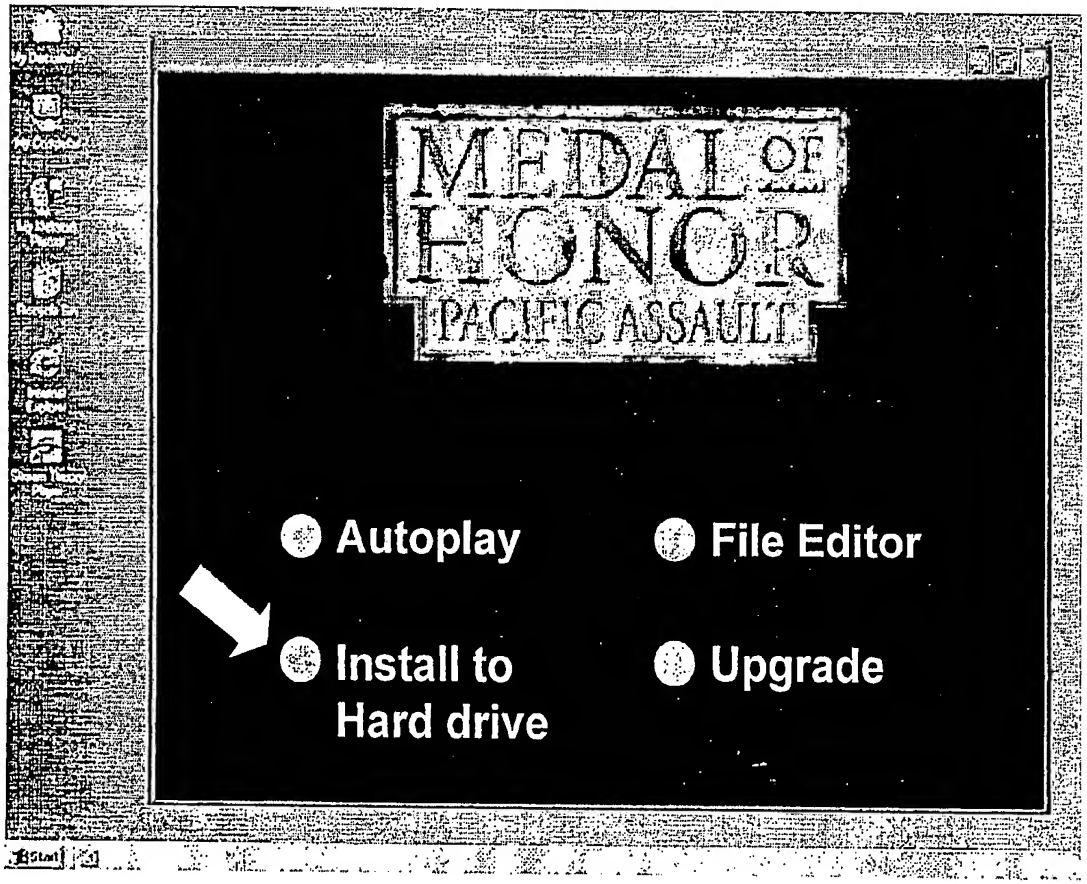


- Game streams from CD
- Game plays using data from CD and "normal" Player cache

FIG. 2G

The *CD Streaming* Experience





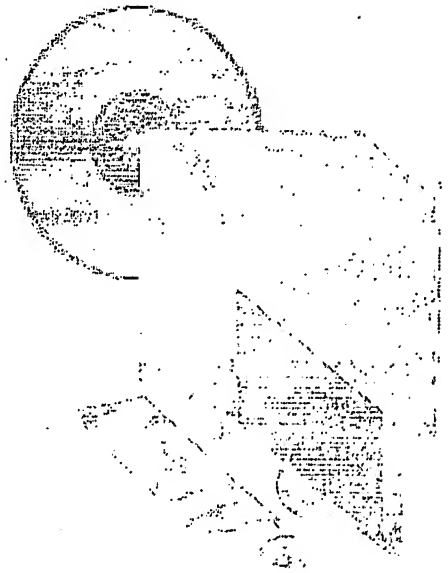
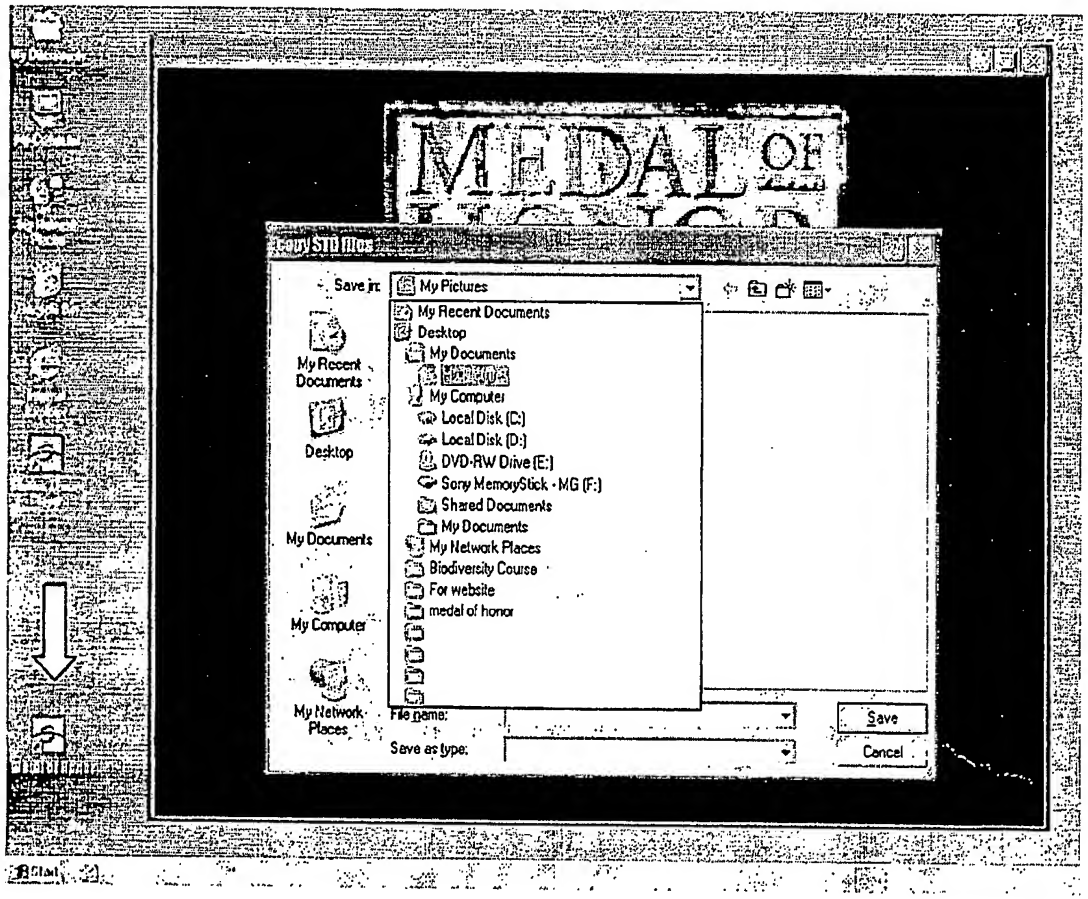
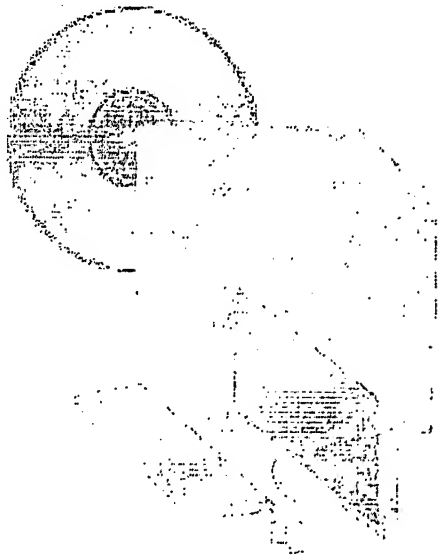


FIG. 2H

The *CD Streaming* Experience

Click on "Install to Hard drive"





Launches program that saves STC files to hard drive

Shortcuts created

Add/Remove capabilities

FIG. 2I

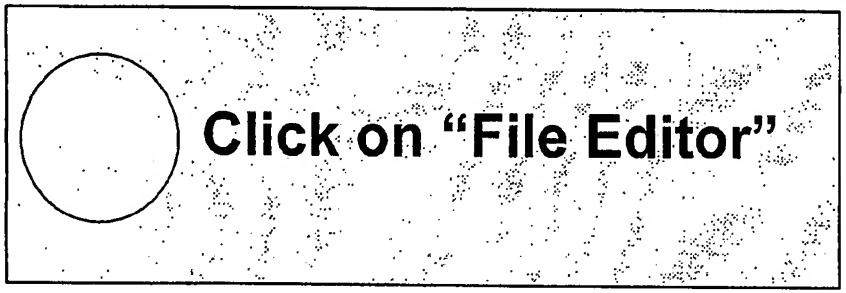
The *CD Streaming* Experience

Click on "Install to Hard drive"

- Select shortcut to Start game
- Launches the Stream Theory Player
- Game streams from Hard disk

FIG. 2J

The *CD Streaming* Experience



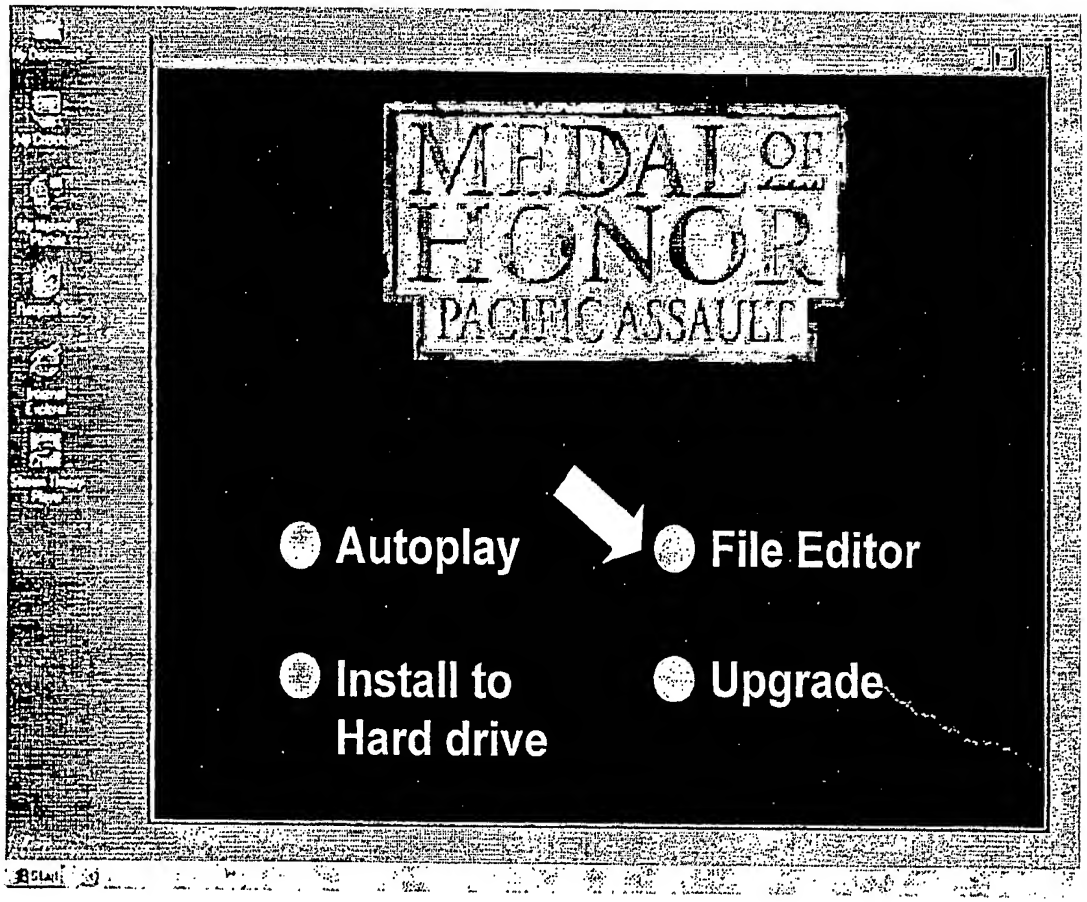
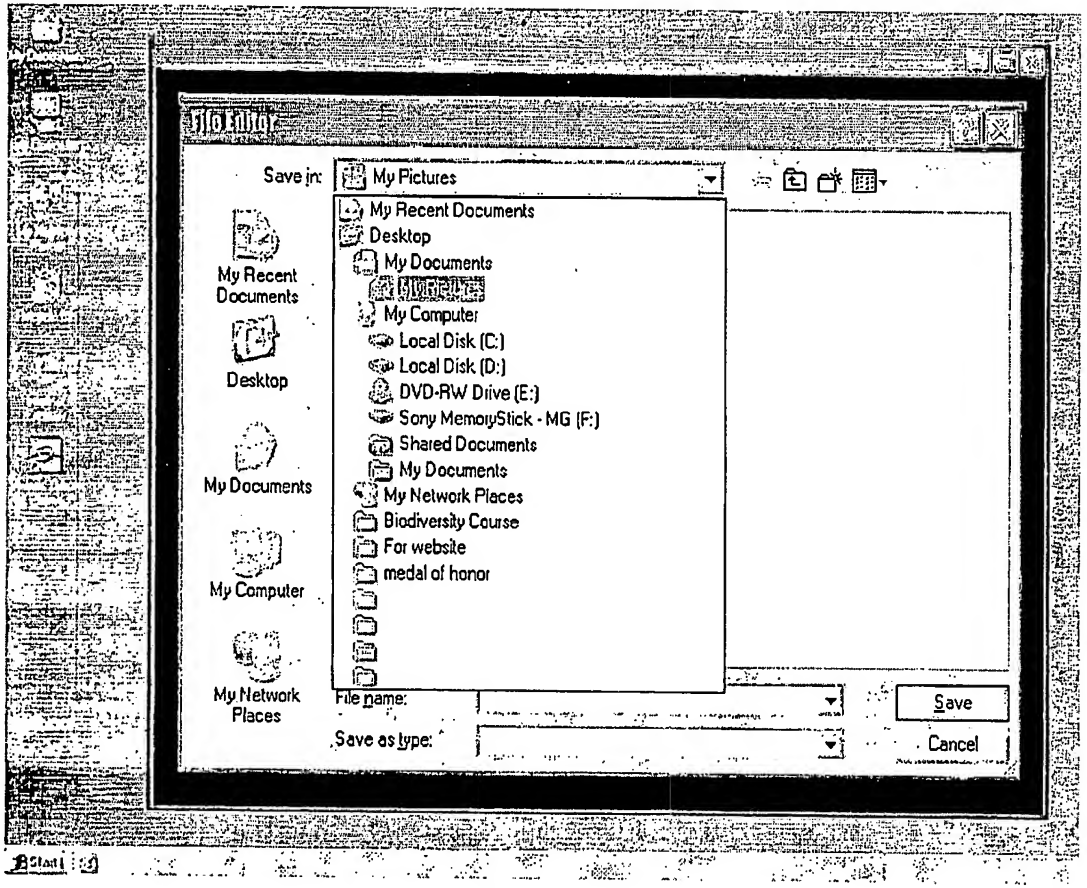


FIG. 2K

The *CD Streaming* Experience

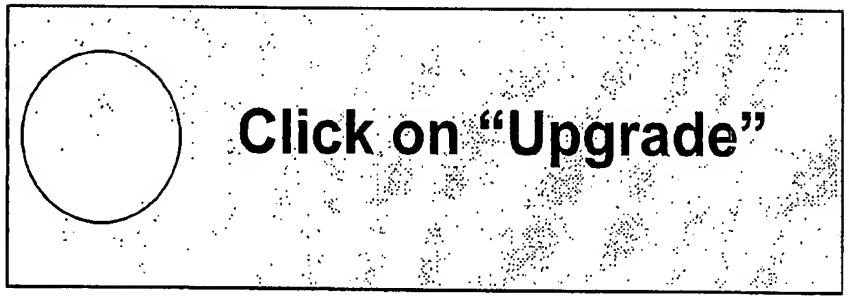
Click on "Install to File Editor"



- Launches File Editor
- User can modify game files
- Automatically saves to "local" subdirectory

FIG. 2L

The *CD Streaming* Experience



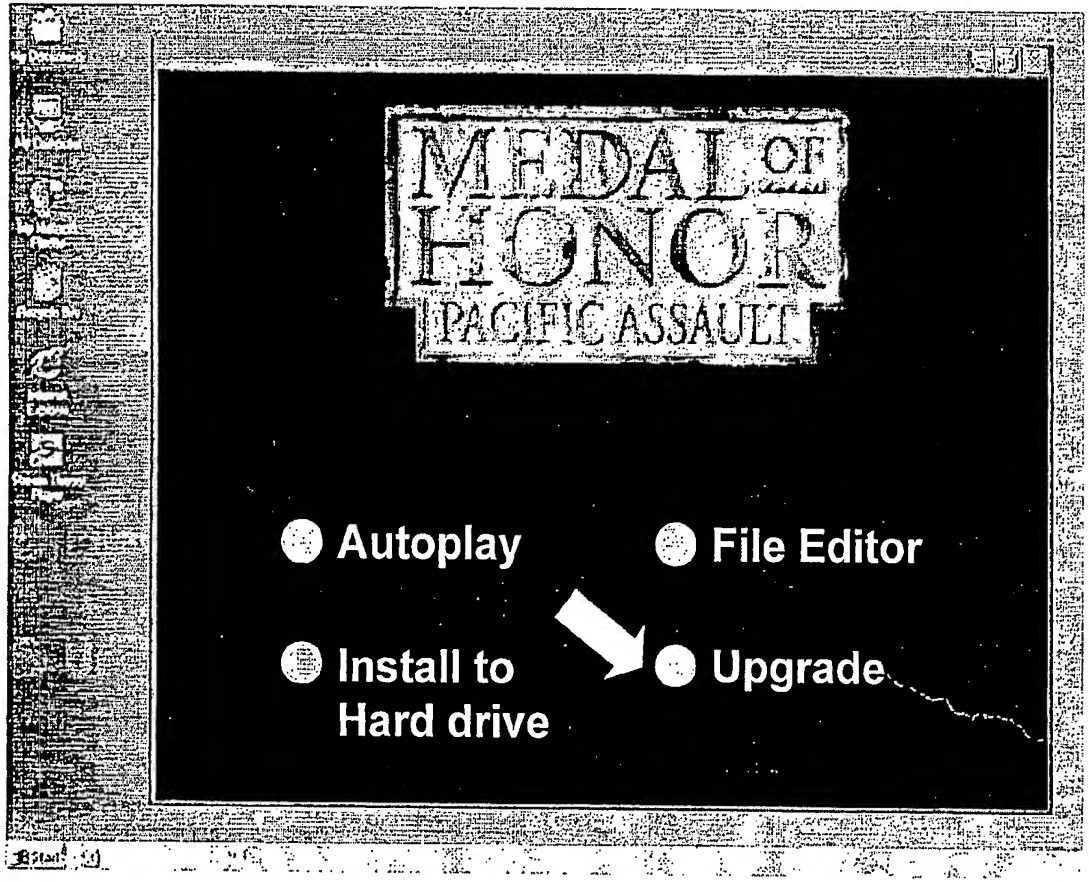
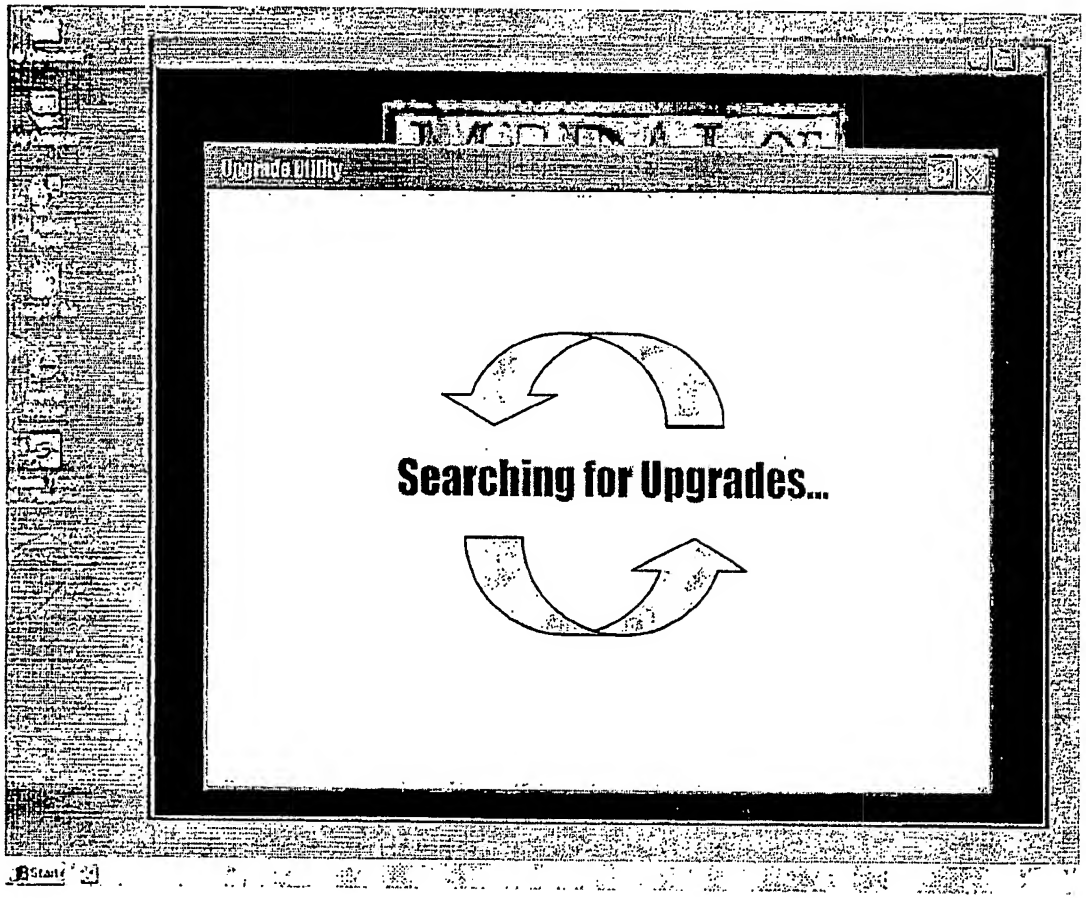


FIG. 2M

The *CD Streaming* Experience

Click on "Upgrade"



- Upgrade program launches to check for updates
- Upgrades applied automatically

FIG. 2N

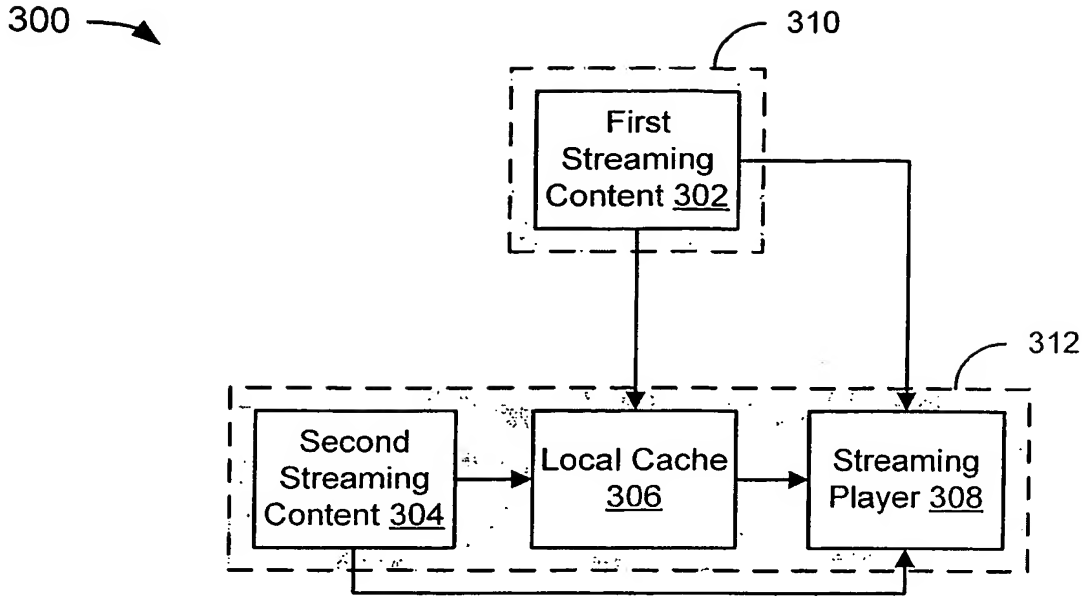

300
310
First Streaming Content 302
312
Second Streaming Content 304
Local Cache 306
Streaming Player 308

FIG. 3A

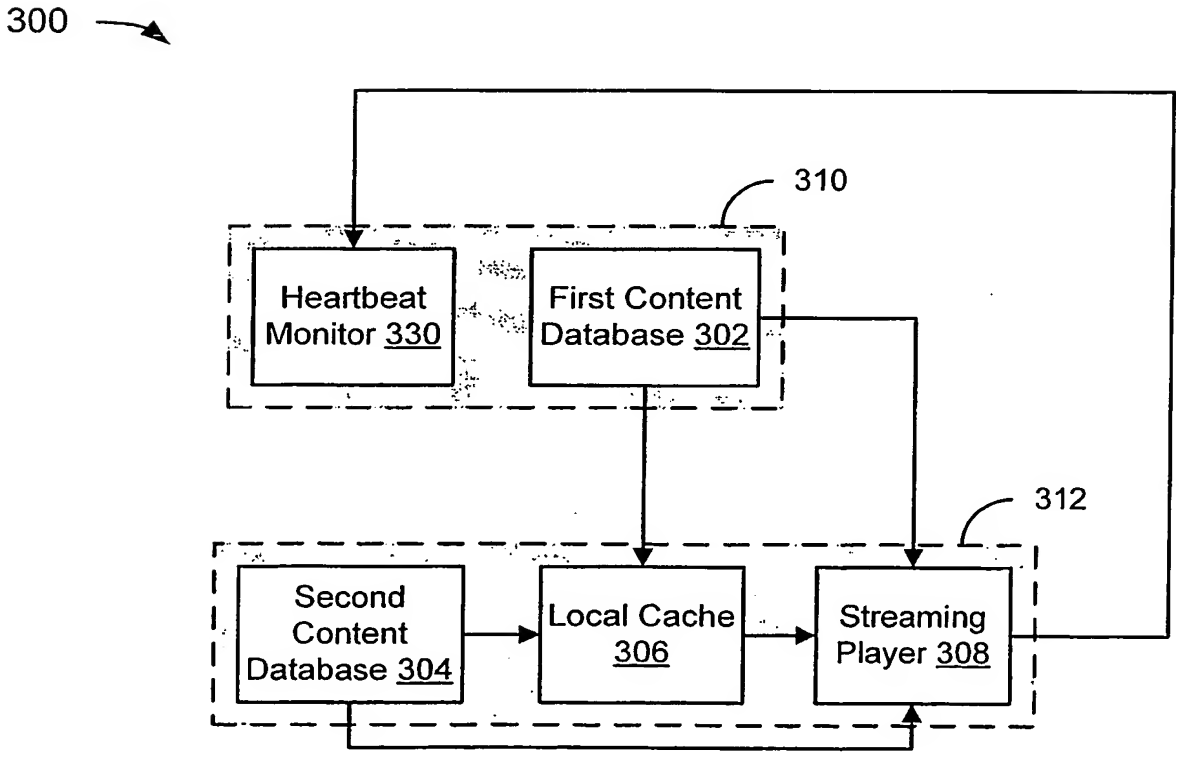

300
310
Heartbeat Monitor 330
First Content Database 302
312
Second Content Database 304
Local Cache 306
Streaming Player 308

FIG. 3B

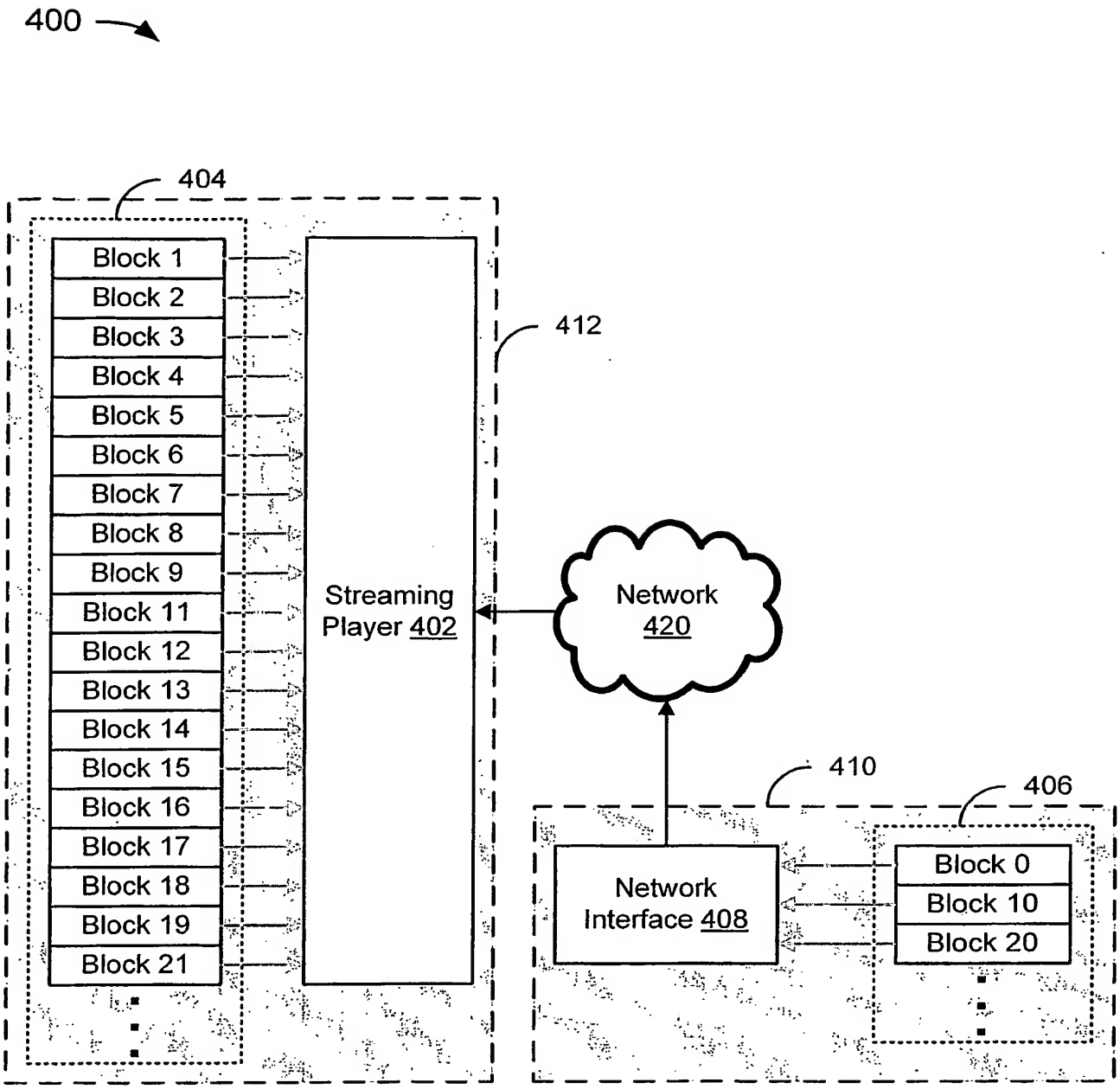

400
404
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 11
Block 12
Block 13
Block 14
Block 15
Block 16
Block 17
Block 18
Block 19
Block 21
412
Streaming Player 402
Network 420
410
406
Network Interface 408
Block 0
Block 10
Block 20

FIG. 4A

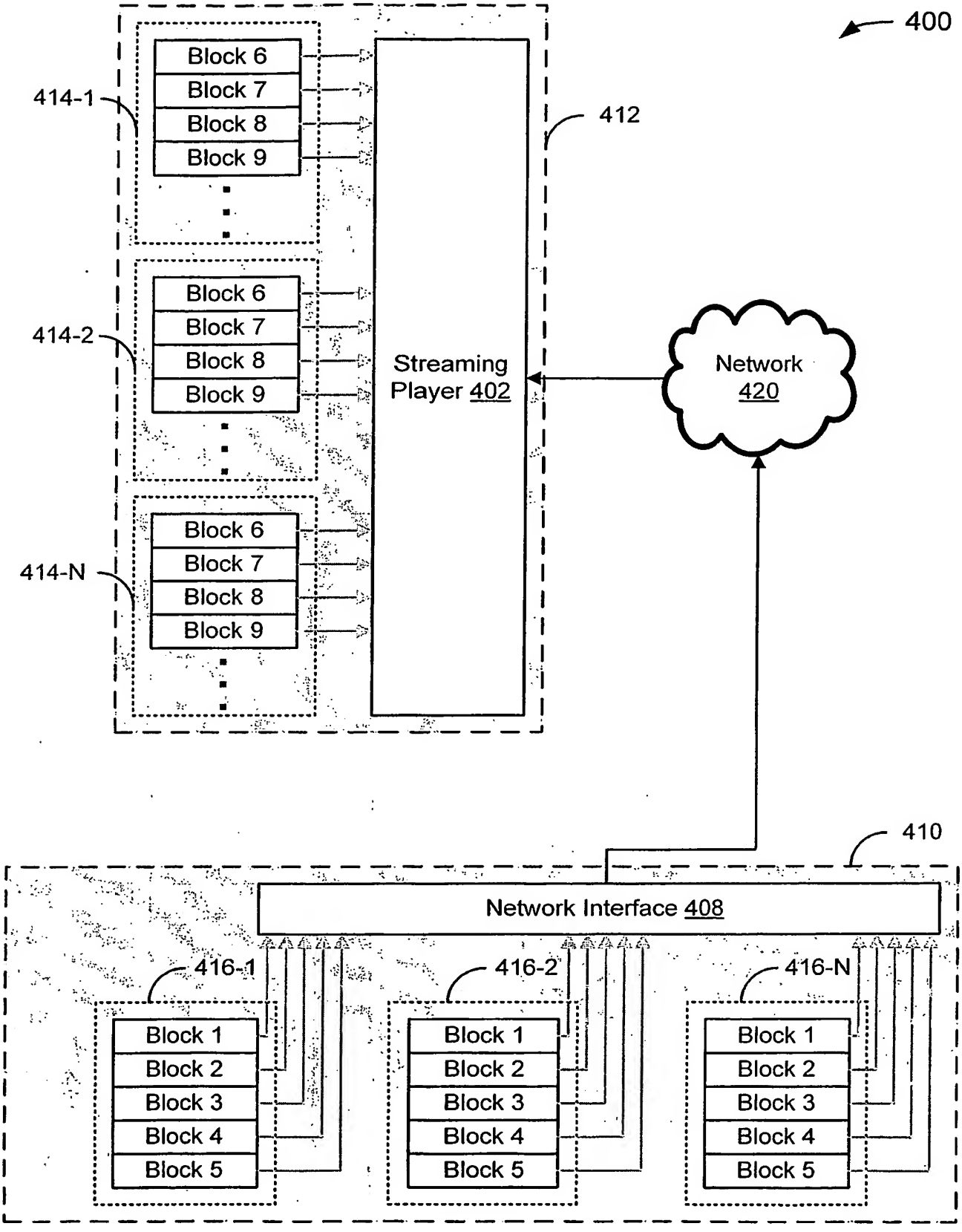

400
414-1
Block 6
Block 7
Block 8
Block 9
412
414-2
Block 6
Block 7
Block 8
Block 9
Streaming Player 402
Network 420
414-N
Block 6
Block 7
Block 8
Block 9
410
Network Interface 408
416-1
Block 1
Block 2
Block 3
Block 4
Block 5
416-2
Block 1
Block 2
Block 3
Block 4
Block 5
416-N
Block 1
Block 2
Block 3
Block 4
Block 5

FIG. 4B

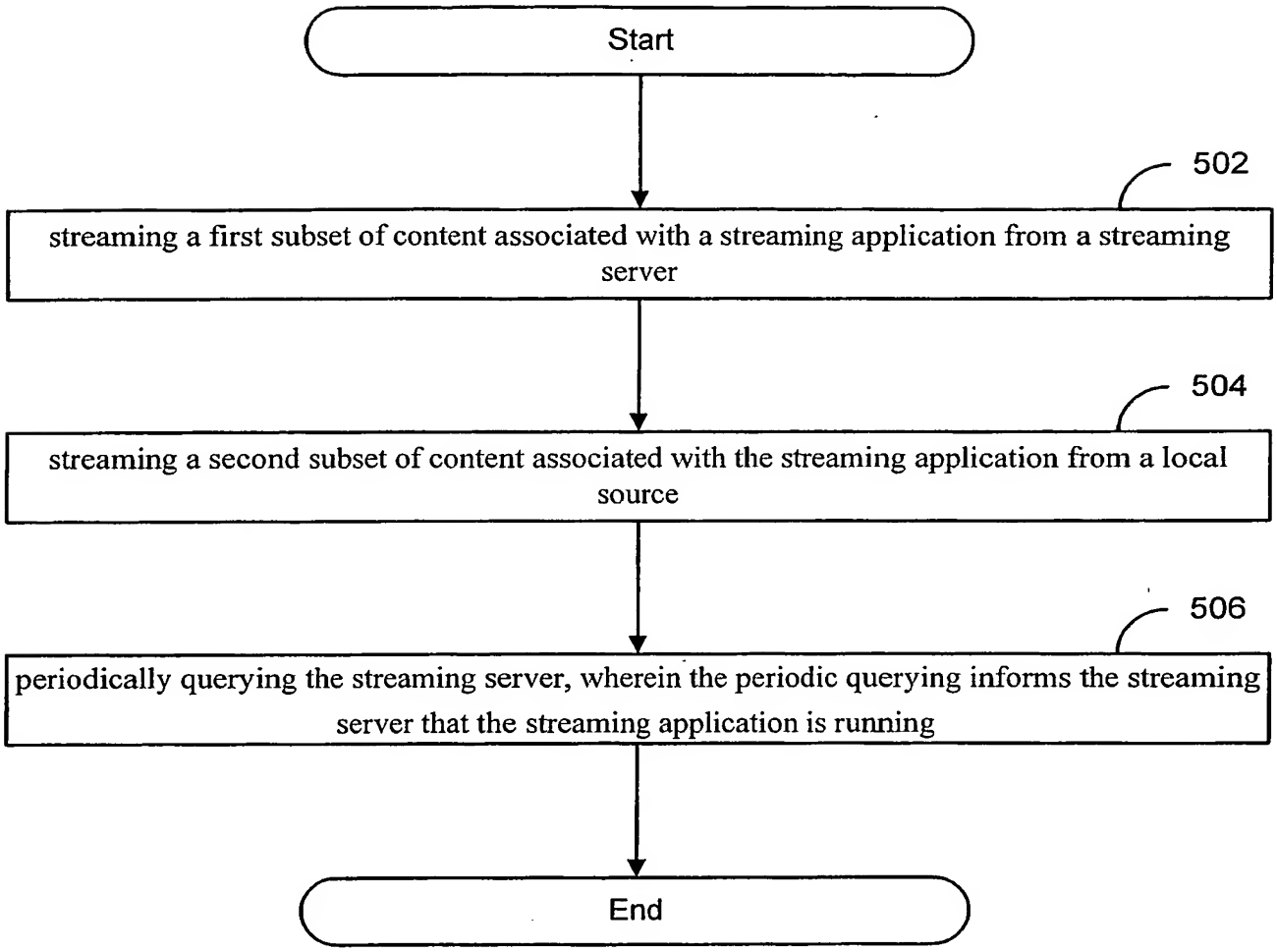

500
Start
502
streaming a first subset of content associated with a streaming application from a streaming server
504
streaming a second subset of content associated with the streaming application from a local source
506
periodically querying the streaming server, wherein the periodic querying informs the streaming server that the streaming application is running
End

FIG. 5

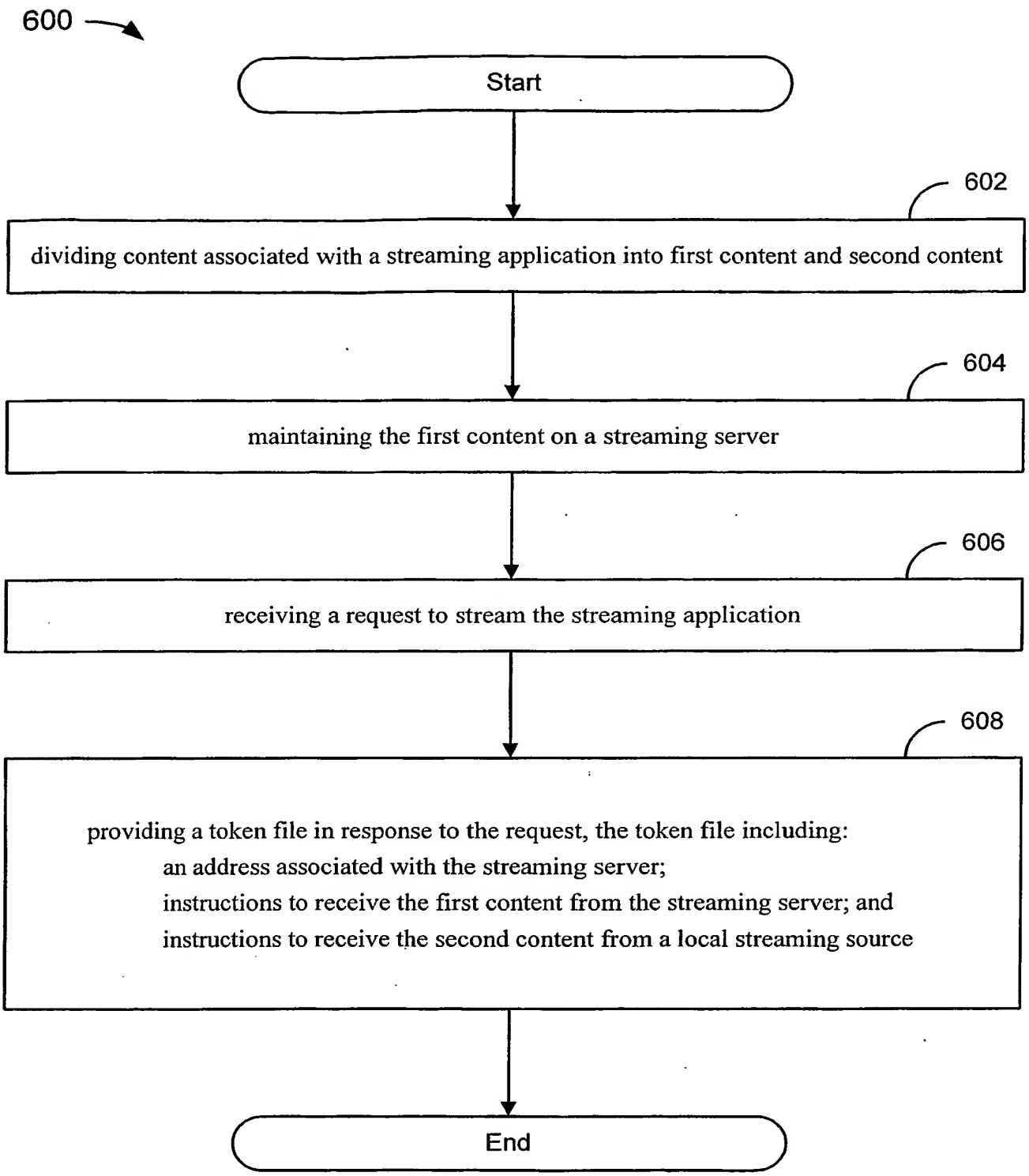

600
Start
602
dividing content associated with a streaming application into first content and second content
604
maintaining the first content on a streaming server
606
receiving a request to stream the streaming application
608
providing a token file in response to the request, the token file including:
an address associated with the streaming server;
instructions to receive the first content from the streaming server; and
instructions to receive the second content from a local streaming source
End

FIG. 6

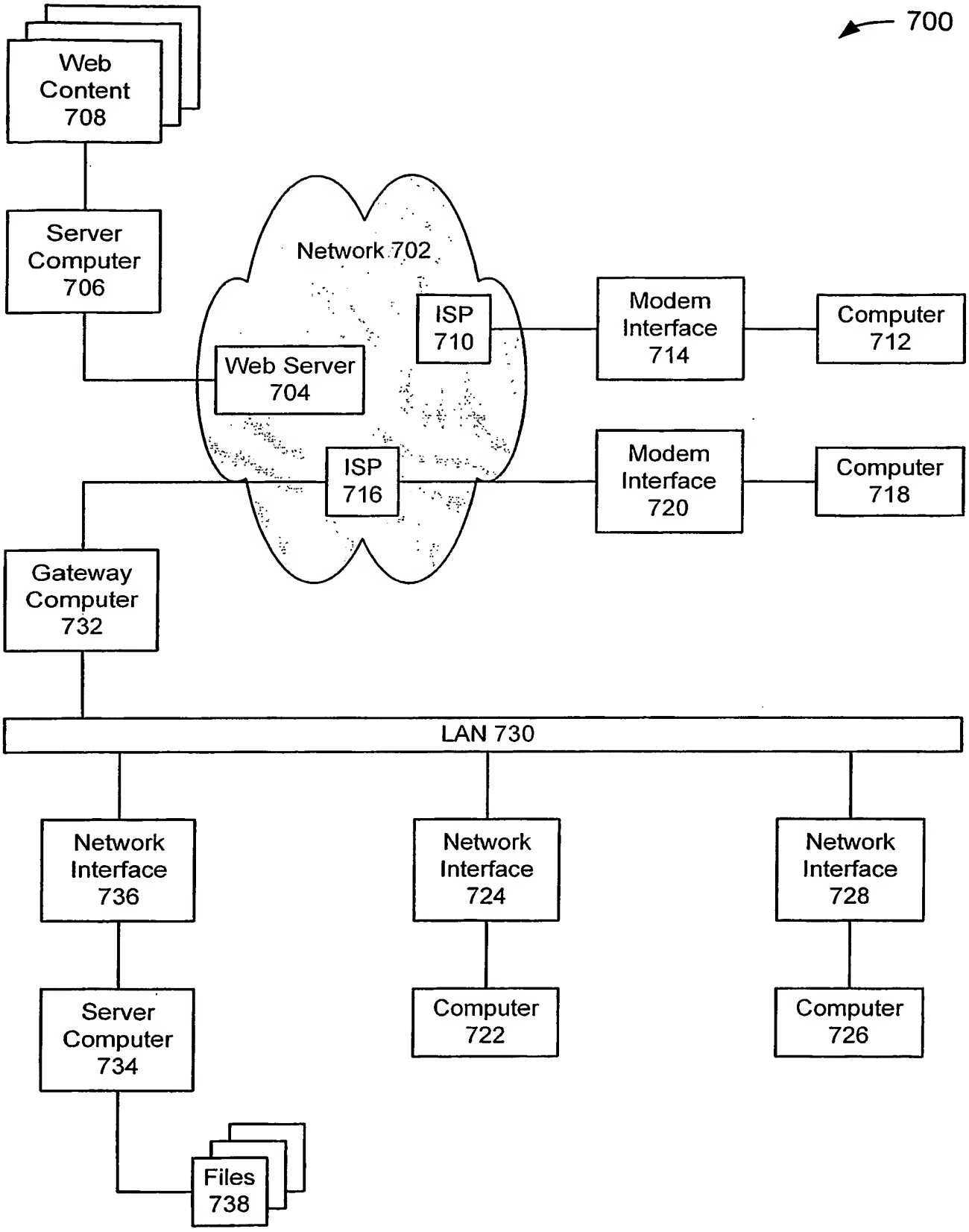

700
Web Content 708
Server Computer 706
Network 702
ISP 710
Web Server 704
ISP 716
Modem Interface 714
Computer 712
Modem Interface 720
Computer 718
Gateway Computer 732
LAN 730
Network Interface 736
Network Interface 724
Network Interface 728
Server Computer 734
Computer 722
Computer 726
Files 738

FIG. 7

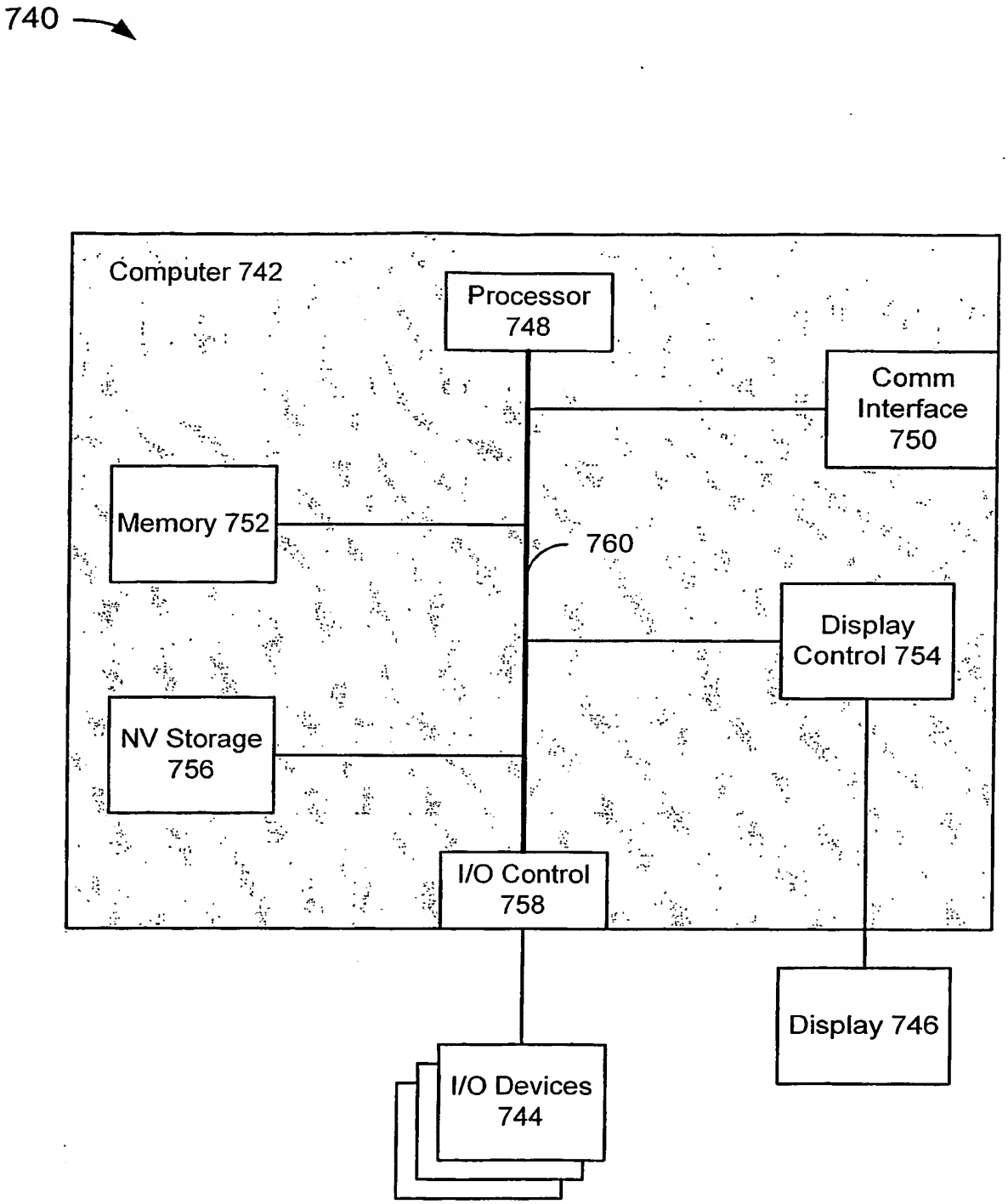

740
Computer 742
Processor 748
Comm Interface 750
Memory 752
760
Display Control 754
NV Storage 756
I/O Control 758
Display 746
I/O Devices 744

FIG. 8